Basic Implementation of Multiple-Interval Pseudospectral Methods to Solve Optimal Control Problems

Technical Report UIUC-ESDL-2015-01

Daniel R. Herber*
Engineering System Design Lab
University of Illinois at Urbana-Champaign

June 4, 2015

Abstract

A short discussion of optimal control methods is presented including indirect, direct shooting, and direct transcription methods. Next the basics of multiple-interval pseudospectral methods are given independent of the numerical scheme to highlight the fundamentals. The two numerical schemes discussed are the Legendre pseudospectral method with LGL nodes and the Chebyshev pseudospectral method with CGL nodes. A brief comparison between time-marching direct transcription methods and pseudospectral direct transcription is presented. The canonical Bryson-Denham state-constrained double integrator optimal control problem is used as a test optimal control problem. The results from the case study demonstrate the effect of user's choice in mesh parameters and little difference between the two numerical pseudospectral schemes.

*Ph.D pre-candidate in Systems and Entrepreneurial Engineering, Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, herber1@illinois.edu

Contents

List of Figures

List of Tables

1 Optimal Control and Direct Transcription

Consider the following general optimal control problem:

$$\min_{\mathbf{u}(t),t_0,t_f} \quad \int_{t_0}^{t_f} \mathcal{L}\left(t,\boldsymbol{\xi}(t),\mathbf{u}(t)\right) dt + \mathcal{M}\left(t_0,\boldsymbol{\xi}(t_0),t_f,\boldsymbol{\xi}(t_f)\right) \tag{1a}$$

$$\text{subject to:} \quad \dot{\boldsymbol{\xi}} - \mathbf{f_d}\left(t,\boldsymbol{\xi}(t),\mathbf{u}(t)\right) = \mathbf{0} \tag{1b}$$

$$\mathbf{C}\left(t,\boldsymbol{\xi}(t),\mathbf{u}(t)\right) \leq \mathbf{0} \tag{1c}$$

$$\boldsymbol{\phi}\left(t_0,\boldsymbol{\xi}(t_0),t_f,\boldsymbol{\xi}(t_f)\right) \leq \mathbf{0} \tag{1d}$$

where t, $\boldsymbol{\xi}(t)$, and $\mathbf{u}(t)$ are the time, state, and control trajectories defined on the time horizon $t \in [t_0, t_f]$. Problem (1) then consists of the Lagrange term $\mathcal{L}(\cdot)$ and Mayer term $\mathcal{M}(\cdot)$ in Eqn. (1a), the dynamic constraints in Eqn. (1b), the algebraic path constraints $\mathbf{C}(\cdot)$ in Eqn. (1c), and the boundary constraints $\boldsymbol{\phi}(\cdot)$ in Eqn. (1d). Time-invariant optimization variables (e.g., plant variables in co-design [Her14a]) may be included as well but are beyond the scope of this report.

Note that Prob. (1) contains $\mathbf{u}(t)$ as an optimization variable. This quantity is infinite-dimensional since it needs to be defined for for all values in $t \in [t_0, t_f]$. Indirect methods of optimal control such as Pontryagin's maximum principle were the initial solution techniques [Lib12, p. 102]. There are a number of issues with using indirect methods (either analytically or numerically), so direct methods are often employed [Her14a, pp. 25–26].

Direct methods of optimal control are known as discretize-then-optimize ($D \rightarrow O$) methods [Bie07, pp. 243–246]. These methods do not use calculus of variations or directly state the optimality conditions. Instead, the control and/or state are parametrized using function approximation and the cost is approximated using numerical quadrature [Rao12, p. 141]. This creates a discrete, finite-dimensional problem that can then solved using large-scale NLP solvers [Bie07, pp. 243–246]. This formulation requires an accurate level of parameterization of the control (and possibly the state) profiles. There are two main classes of direct methods: sequential and simultaneous. Sequential methods only parametrize the control while simultaneous methods parametrize both the state and control. In both methods, the parameters that define the parameterization are included as optimization variables. Sequential methods include both shooting and multiple shooting formulations which utilize a nested differential-algebraic equation solver (simulation) to satisfy the dynamic constraints. The simultaneous approach on the other hand simultaneously searches for the optimal solution and feasible dynamic constraints. Sequential approaches typically produce low-quality solutions and are computationally inefficient, i.e., compared to simultaneous methods [Her14a, pp. 26–28].

The simultaneous or direct transcription (DT) problem formulation is:

$$\min_{\boldsymbol{\Xi},\mathbf{U},t_0,t_f} \quad \sum_{k=0}^{N_t} w_k \mathcal{L}\left(t_k,\boldsymbol{\xi}(t_k),\mathbf{u}(t_k)\right) + \mathcal{M}\left(t_0,\boldsymbol{\xi}(t_0),t_f,\boldsymbol{\xi}(t_f)\right) \tag{2a}$$

$$\text{subject to:} \quad \boldsymbol{\zeta}\left(\mathbf{t},\boldsymbol{\Xi},\mathbf{U}\right) = \mathbf{0} \tag{2b}$$

$$\mathbf{C}\left(\mathbf{t},\boldsymbol{\Xi},\mathbf{U}\right) \leq \mathbf{0} \tag{2c}$$

$$\boldsymbol{\phi}\left(t_0,\boldsymbol{\xi}(t_0),t_f,\boldsymbol{\xi}(t_f)\right) \leq \mathbf{0} \tag{2d}$$

where N_t+1 is the number of discrete time points and $\mathbf{t}$, $\boldsymbol{\Xi}$, and $\mathbf{U}$ are the discretized forms of the time, states, and controls. The Lagrange term is approximated with numerical quadrature with weights w_k. The dynamic constraint is now enforced through a large number of equality constraints $\boldsymbol{\zeta}(\cdot)$, termed defect constraints. DT approaches can either have either local or global support termed time-marching methods or pseudospectral methods, respectively [Her14a, p. 29]. The primary focus of this technical report is pseudospectral methods.

2 Basics of Pseudospectral Methods

Here we discuss how to approximate the infinite-dimensional Prob. (1) as the finite-dimensional Prob. (2) with pseudospectral (PS) methods, one class of direct transcription methods. PS methods parametrize both the control and state using function approximation and the cost using numerical quadrature [Rao12, p. 141]. Some general references on various types of PS methods include Refs. [FR02, FR08, BGa10, Rao12]. A number of software packages exist to implement PS methods including $\mathbb{GPOPS}-\mathbb{II}$ [PR14], $\mathcal{PSOPT}$ [Bec11], and PROPT [RE10].

Section 2.1 provides the foundational elements for PS methods independent of particular choice of nodes, basis function, etc. for a single polynomial defined over the time horizon. Next in Sec. 2.2 we extend the PS approach to multiple-intervals each containing a distinct polynomial approximation. In Secs. 2.3 and 2.4 we outline the nodes, basis function, etc. for two particular numerical schemes. Finally, Sec. 2.5 briefly compares PS methods to the time-marching DT methods.

2.1 Foundation

2.1.1 Transformation to τ

We define a new independent, scaled time variable τ instead of t as:

$$\tau = \frac{2}{t_{N_t} - t_0} t - \frac{t_{N_t} + t_0}{t_{N_t} - t_0} \tag{3}$$

where $\tau \in [-1, 1]$ for any value of t_0 and $t_{N_t} \equiv t_f$.

2.1.2 Interpolation with Polynomials

Given a set of node points $\boldsymbol{\tau} = [\tau_0, \tau_1, \ldots, \tau_{N_t}]^T$ containing N_t+1 increasing distinct numbers in $[-1, 1]$ and $f(\tau)$ is a function whose values are given at $\boldsymbol{\tau}$, then a unique polynomial $P(\tau)$ of degree at most N_t exists with:

$$f(\tau_k) = P(\tau_k), \qquad k = \{0, 1, \ldots, N_t\} \tag{4}$$

The interpolating polynomial of $f(\tau)$ is defined as:

$$f(\tau) \approx P(\tau) = \sum_{k=0}^{N_t} f(\tau_k)\phi_k(\tau) \tag{5}$$

where $\phi_k(\cdot)$ are continuous basis polynomials that are typically constructed such that the property in Eqn. (4) holds, i.e., interpolating $f(\tau)$ at the node points. Therefore, the polynomial approximation is exact at the node points and an approximation at all other values of τ.

2.1.3 Approximate Differentiation

The derivative $(d/d\tau)$ of $f(\tau)$ can be approximated with $dP(\tau)/d\tau$:

$$\dot{f}(\tau_k) \approx \dot{P}(\tau_k) = \sum_{i=0}^{N_t} D_{ki} f(\tau_i) \tag{6}$$

where the $(N_t + 1) \times (N_t + 1)$ matrix $\mathbf{D}$ is termed the differentiation matrix. This matrix depends only on the values of $\boldsymbol{\tau}$ and the type of interpolating polynomial, so we have an approximation for the $\dot{f}(\boldsymbol{\tau})$ that depends only on $f(\boldsymbol{\tau})$.

2.1.4 Numerical Quadrature

We first note that the transformation in Eqn. (3) scales the integral term:

$$\int_{t_0}^{t_{N_t}} f(t)dt = \frac{t_{N_t} - t_0}{2} \int_{-1}^{1} f(\tau)d\tau \tag{7}$$

For a function $f(\tau)$, the integral over $\tau \in [-1, 1]$ can be approximated with a quadrature scheme:

$$\int_{-1}^{1} f(\tau) \approx \sum_{k=0}^{N_t} w_k f(\tau_k) \tag{8}$$

where w_k are predetermined quadrature weights. These weights depend only on the values of $\boldsymbol{\tau}$ and the type of interpolating polynomial, so we have an approximation of the definite integral that is a sum of predefined weighted values of $f(\boldsymbol{\tau})$.

2.1.5 Discretization Matrices and Optimization Variable Vector

For simplicity, we denote the derivative function and the time step of the entire time horizon as:

$$\mathbf{f_d}(\tau) \rightleftharpoons \mathbf{f_d}\left(\tau, \boldsymbol{\xi}(\tau), \mathbf{u}(\tau)\right) \tag{9}$$

$$h = t_{N_t} - t_0 \tag{10}$$

We define the following matrices that contain the discreztized components of Prob. (1):

$$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}(\tau_0) \\ \vdots \\ \boldsymbol{\xi}(\tau_{N_t}) \end{bmatrix} = \begin{bmatrix} \xi_1(\tau_0) & \cdots & \xi_{n_\xi}(\tau_0) \\ \vdots & \ddots & \vdots \\ \xi_1(\tau_{N_t}) & \cdots & \xi_{n_\xi}(\tau_{N_t}) \end{bmatrix}_{(N_t+1)\times n_\xi} \tag{11a}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}(\tau_0) \\ \vdots \\ \mathbf{u}(\tau_{N_t}) \end{bmatrix} = \begin{bmatrix} u_1(\tau_0) & \cdots & u_{n_u}(\tau_0) \\ \vdots & \ddots & \vdots \\ u_1(\tau_{N_t}) & \cdots & u_{n_u}(\tau_{N_t}) \end{bmatrix}_{(N_t+1)\times n_u} \tag{11b}$$

$$\mathbf{F} = \frac{h}{2} \begin{bmatrix} \mathbf{f_d}(\tau_0) \\ \vdots \\ \mathbf{f_d}(\tau_{N_t}) \end{bmatrix} = \frac{h}{2} \begin{bmatrix} f_{d1}(\tau_0) & \cdots & f_{dn_\xi}(\tau_0) \\ \vdots & \ddots & \vdots \\ f_{d1}(\tau_{N_t}) & \cdots & f_{dn_\xi}(\tau_{N_t}) \end{bmatrix}_{(N_t+1)\times n_\xi} \tag{11c}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}(\tau_0) \\ \vdots \\ \mathbf{C}(\tau_{N_t}) \end{bmatrix} = \begin{bmatrix} C_1(\tau_0) & \cdots & C_{n_C}(\tau_0) \\ \vdots & \ddots & \vdots \\ C_1(\tau_{N_t}) & \cdots & C_{n_C}(\tau_{N_t}) \end{bmatrix}_{(N_t+1)\times n_C} \tag{11d}$$

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_1 & \cdots & \phi_{n_\phi} \end{bmatrix}_{1\times n_\phi} \tag{11e}$$

where n_ξ, n_u, n_C, and n_ϕ are the number of states, controls, path constraints, and boundary constraints, respectively. We briefly note here that $\mathbf{C} \leq \mathbf{0}$ and $\boldsymbol{\phi} \leq \mathbf{0}$ now can be included directly in the NLP formulation although constraint satisfaction of $\mathbf{C}$ is only at the finite set of node points. In addition, the optimization variable vector dimension scales linearly with N_t:

$$\mathbf{x} = \begin{bmatrix} \text{vec}(\boldsymbol{\Xi}) \\ \text{vec}(\mathbf{U}) \\ t_0 \\ t_f \end{bmatrix}_{(N_t+1)\times(n_\xi+n_u)+2} \tag{12}$$

where the function vec$(\cdot)$ reorganizes the matrix input to a single column vector containing all entries of the input matrix.

2.1.6 Defect Constraints

To represent accurately the dynamics in Prob. (1b), we need to ensure that our approximation for the state derivatives using Eqn. (6) is equivalent to the derivative function values given by $\mathbf{f_d}(\tau)$. This enforced with equality constraints in matrix form:

$$\begin{aligned} \boldsymbol{\zeta} &= \mathbf{0} \\ &= \mathbf{D}\boldsymbol{\Xi} - \mathbf{F} \end{aligned} \tag{13}$$

recalling that $\mathbf{D}$ is the differentiation matrix defined in Eqn. (6), $\boldsymbol{\Xi}$ is the matrix of discretized state values defined in Eqn. (11a), and $\mathbf{F}$ is the matrix of derivative function values defined in Eqn. (11c).

2.1.7 Objective Function

The definite integral with the Lagrange term in Eqn. (2a) can be approximated by:

$$\sum_{k=0}^{N_t} w_k \mathcal{L}\left(t_k, \boldsymbol{\xi}(t_k), \mathbf{u}(t_k)\right) = \frac{h}{2} \sum_{k=0}^{N_t} w_k \mathcal{L}\left(\tau_k, \boldsymbol{\xi}(\tau_k), \mathbf{u}(\tau_k)\right) \tag{14}$$

For simplicity, we denote the Lagrange term as:

$$\mathcal{L}\left(\tau_k\right) \rightleftharpoons \mathcal{L}\left(\tau_k, \boldsymbol{\xi}(\tau_k), \mathbf{u}(\tau_k)\right) \tag{15}$$

The Mayer term is simply:

$$\mathcal{M}\left(t_0, \boldsymbol{\xi}(t_0), t_{N_t}, \boldsymbol{\xi}(t_{N_t})\right) = \mathcal{M}\left(t_0, \boldsymbol{\xi}(-1), t_{N_t}, \boldsymbol{\xi}(1)\right) \tag{16}$$

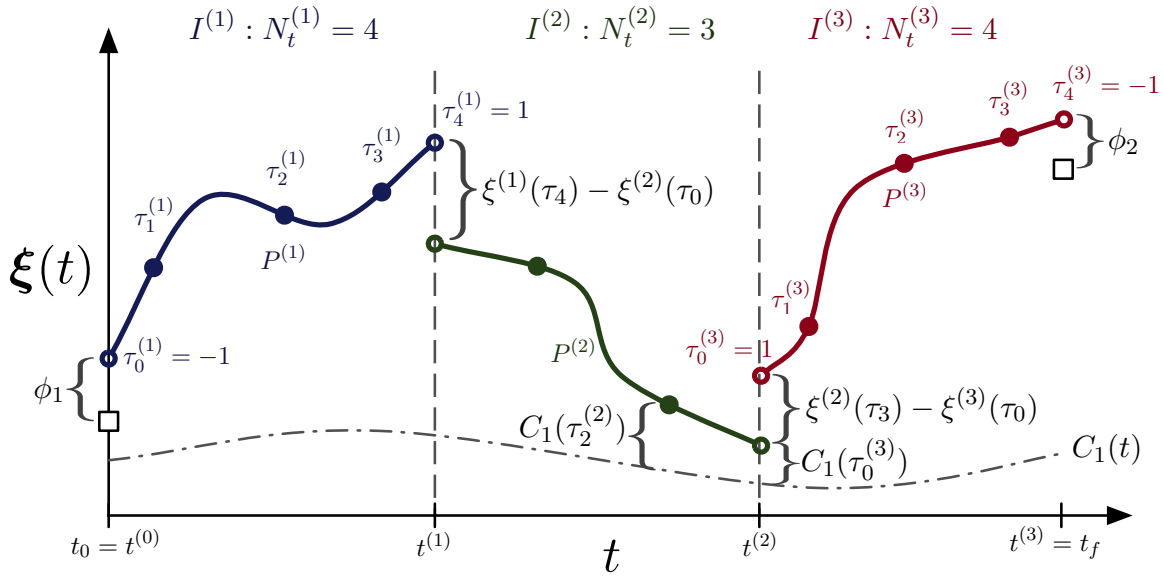


Figure 1 Illustration of a state variable in multiple-interval psuedospectal methods.

2.2 Multiple Intervals

It may be beneficial to use multiple polynomial approximations over the t_0 to t_{N_t} time horizon since the true solution may not be accurately approximated with a single polynomial [Rao12]. This approach is illustrated in Fig. 1.

2.2.1 Mesh Intervals

The ith mesh interval is denoted $I^{(i)}$ where $i = \{1, 2, \ldots, N_I\}$. Each mesh interval contains a polynomial $P^{(i)}$ of degree $N_t^{(i)}$ in the time interval $[t^{(i-1)}, t^{(i)}]$ where:

$$t_0 = t^{(0)} < t^{(1)} < t^{(2)} < \cdots < t^{(N_I)} = t_f \tag{17}$$

Each time interval will still be scaled according to Eqn. (3) so that $[t^{(i-1)}, t^{(i)}] \rightarrow [-1, 1]$. Let us denote the set of state and control discretization matrices in all of the mesh intervals as:

$$\mathcal{S} = \left\{ \boldsymbol{\Xi}^{(i)}, i = \{1, 2, \ldots, N_I\} \right\}, \qquad \mathcal{U} = \left\{ \mathbf{U}^{(i)}, i = \{1, 2, \ldots, N_I\} \right\} \tag{18}$$

2.2.2 Continuity Constraint

An additional constraint must be added to ensure continuity in state between the mesh intervals:

$$\boldsymbol{\xi}^{(i-1)}(\tau_{N_t^{(i-1)}}) = \boldsymbol{\xi}^{(i)}(\tau_0), \qquad i = \{2, 3, \ldots, N_I\} \tag{19}$$

where we note that there are $(N_I - 1) \times n_\xi$ continuity constraints. Put into words, final states of the left interval must equal the initial states of the right interval for continuity between polynomial approximations.

2.2.3 Multiple-Interval Formulation

The complete multiple-interval formulation of Prob. (2) is:

$$\min_{\mathcal{S},\mathcal{U},t_0,t_f} \quad \sum_{i=1}^{N_I}\left[\frac{h^{(i)}}{2}\sum_{k=0}^{N_t^{(i)}} w_k^{(i)}\mathcal{L}\left(\tau_k^{(i)}\right)\right] + \mathcal{M}\left(t_0, \boldsymbol{\xi}^{(1)}(-1), t_f, \boldsymbol{\xi}^{(N_I)}(1)\right) \tag{20a}$$

$$\text{subject to:} \quad \boldsymbol{\zeta}\left(\boldsymbol{\tau}^{(i)}, \boldsymbol{\Xi}^{(i)}, \mathbf{U}^{(i)}\right) = \mathbf{0} \qquad i = \{1, 2, \ldots, N_I\} \tag{20b}$$

$$\mathbf{C}\left(\boldsymbol{\tau}^{(i)}, \boldsymbol{\Xi}^{(i)}, \mathbf{U}^{(i)}\right) \leq \mathbf{0} \qquad i = \{1, 2, \ldots, N_I\} \tag{20c}$$

$$\boldsymbol{\phi}\left(t_0, \boldsymbol{\xi}^{(1)}(-1), t_f, \boldsymbol{\xi}^{(N_I)}(1)\right) \leq \mathbf{0} \tag{20d}$$

$$\boldsymbol{\xi}^{(i-1)}(\tau_{N_t^{(i-1)}}) - \boldsymbol{\xi}^{(i)}(\tau_0) = \mathbf{0} \qquad i = \{2, 3, \ldots, N_I\} \tag{20e}$$

The problem size now additionally depends on the number of intervals N_I and their respective number of nodes points $N_t^{(i)}$.

2.3 Legendre Pseudospectral Method with LGL Nodes

This section outlines one particular numerical scheme for finding the nodes, differentiation matrix, and quadrature weights for use in pseudospectral methods.

2.3.1 Nodes

Let $L_N(\tau)$ denote the Legendre polynomial of order N, which may be generated from:

$$L_N(\tau) = \frac{1}{2^N N!} \frac{d^N}{d\tau^N} \left(\tau^2 - 1\right)^N \tag{21}$$

The Lagrange-Gauss-Lobatto (LGL) nodes are defined as:

$$\tau_k = \begin{cases} -1 & \text{if } k = 0 \\ k\text{th root of } \dot{L}_{N_t}(\tau) & \text{if } k = \{1, 2, \ldots, N_t - 1\} \\ 1 & \text{if } k = N_t \end{cases} \tag{22}$$

where $\dot{L}_N = \frac{dL_N}{d\tau}$. We note that the nodes are always between $[-1, 1]$ and contain both endpoints (App. C code from Ref. [STW11]).

2.3.2 Interpolating Polynomial Basis Function

We define the basis polynomials needed in Eqn. (5) for the Legendre-based method as Lagrange basis polynomials:

$$\phi_k(\tau) = \prod_{i=0, i\neq k}^{N_t} \frac{\tau - \tau_i}{\tau_k - \tau_i} \tag{23}$$

With LGL nodes, $\phi_k(\tau)$ can be written in the following alternative form [BGa10]:

$$\phi_k(\tau) = \frac{1}{N_t \left(N_t + 1\right) L_{N_t}(\tau_k)} \frac{(\tau^2 - 1)\dot{L}_{N_t}(\tau)}{\tau - \tau_k} \tag{24}$$

2.3.3 Approximate Differentiation

The differentiation matrix needed in Eqn. (6) for the Legendre-based method is:

$$D_{ki} = \begin{cases} \frac{L_{N_t}(\tau_k)}{L_{N_t}(\tau_i)} \frac{1}{\tau_k - \tau_i} & \text{if } k \neq i \\ N_t(N_t + 1)/4 & \text{if } k = i = 0 \\ -N_t(N_t + 1)/4 & \text{if } k = i = N_t \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

Further numerical enhancements can be made to improve stability in the presence of rounding errors (expression from Ref. [BGa10], App. C code from Ref. [STW11]).

2.3.4 Quadrature Weights

The quadrature weights w_k needed in Eqn. (8) for the Legendre-based method are:

$$w_k = \frac{2}{N_t(N_t + 1)} \frac{1}{\left(L_{N_t}(\tau_k)\right)^2}, \qquad k = \{0, 1, \ldots, N_t\} \tag{26}$$

These are Gaussian quadrature weights that are exactly accurate for polynomials of degree up to degree $2N_t - 1$ (expression from Ref. [FR08], App. C code from Ref. [STW11]).

2.4 Chebyshev Pseudospectral Method with CGL Nodes

This section outlines one particular numerical scheme for finding the nodes, differentiation matrix, and quadrature weights for use in pseudospectral methods.

2.4.1 Nodes

Let $T_N(\tau)$ denote the Chebyshev polynomial of order N, which may be generated from:

$$T_N = \cos\left(N \cos^{-1}(\tau)\right) \tag{27}$$

The Chebyshev-Gauss-Lobatto (CGL) nodes are defined as the roots of $\dot{T}_{N_t} = \frac{dT_{N_t}}{d\tau}$ and the additional endpoints. All CGL nodes can be computed conveniently by:

$$\tau_k = -\cos\left(\frac{\pi k}{N_t}\right) \qquad k = \{0, 1, \ldots, N_t\} \tag{28}$$

We note that the nodes are always between $[-1, 1]$ and contain both endpoints.

2.4.2 Interpolating Polynomial Basis Function

We define the basis polynomials needed in Eqn. (5) for the Chebyshev-based method as Lagrange basis polynomials previously defined in Eqn. (23). With CGL nodes, $\phi_k(\tau)$ can be written in the following alternative form [FR02]:

$$\phi_k(\tau) = \frac{(-1)^{k+1}}{N_t^2 a_k} \frac{(1-\tau^2)\dot{T}_{N_t}(\tau)}{\tau - \tau_k} \quad \text{where: } a_k = \begin{cases} 2 & \text{if } k = \{0, N_t\} \\ 1 & \text{otherwise} \end{cases} \tag{29}$$

2.4.3 Approximate Differentiation

The differentiation matrix needed in Eqn. (6) for the Chebyshev-based method is:

$$D_{ki} = \begin{cases} \frac{a_k}{a_i} \frac{(-1)^{k+i}}{(\tau_k - \tau_i)} & \text{if } k \neq i \\ -\frac{\tau_k}{2(1-\tau_k^2)} & \text{if } 1 \leq k = i \leq N_t - 1 \\ \frac{2N_t^2+1}{6} & \text{if } k = i = 0 \\ -\frac{2N_t^2+1}{6} & \text{if } k = i = N_t \end{cases} \tag{30}$$

Further numerical enhancements can be made to improve stability in the presence of rounding errors (expression from Ref. [FR02], App. C code from [Tre00, p. 54]).

2.4.4 Quadrature Weights

The quadrature weights w_k needed in Eqn. (8) for the Chebyshev-based method are:

$$w_k = \frac{c_k}{N_t}\left(1 - \sum_{j=1}^{\lfloor N_t/2 \rfloor} \frac{b_j}{4j^2 - 1} \cos(2j\tau_k)\right) \tag{31}$$

$$\text{where: } b_j = \begin{cases} 1 & \text{if } j = N_t/2 \\ 2 & \text{if } j < N_t/2 \end{cases}, \qquad c_k = \begin{cases} 1 & \text{if } k = \{0, N_t\} \\ 2 & \text{otherwise} \end{cases}$$

These are Clenshaw-Curtis quadrature weights that are exactly accurate for polynomials of degree up to degree N_t (expression from Ref. [Wal06], App. C code from Ref. [Tre00, p. 128]).

2.5 Brief Comparison to Time-Marching Direct Transcription Methods

2.5.1 Foundation

Time-marching DT methods approach the satisfaction of the differential equation in Eqn. (2b) in a slightly different manner than PS methods although both can produce approximately feasible dynamics when the defect constraints are satisfied.

Consider the following integral equation that provides a solution to $\dot{\boldsymbol{\xi}} = \mathbf{f_d}(\cdot)$ for a given initial condition over the $t \in [t_0, t_f]$ time horizon:

$$\boldsymbol{\xi}(t_f) = \boldsymbol{\xi}(t_0) + \int_{t_0}^{t_f} \mathbf{f_d}(s, \boldsymbol{\xi}(s), \mathbf{u}(s))\, ds \tag{32}$$

A common approach for approximating this type of equation is the class of Runge-Kutta methods which only directly require the knowledge of the state and control values at the interval boundaries [Bet10, pp. 97–100]. However, these methods are only accurate with a sufficiently small step sizes. Therefore we will take the same approach as Sec. 2.2: multiple-intervals.

Reconsider Eqn. (17) which defined the temporal relationship between the time interval boundaries. If we take $N_t^{(i)} = 1 \ \forall i \in \{1, 2, \ldots, N_I\}$, then each interval would only contain the scaled points -1 and 1. Based this structure, we will forgo the continuity equation in Eqn. (19) and instead use a single value for the final point of the $I^{(i-1)}$ and initial point of $I^{(i)}$. Therefore we will have a $N_t = N_I$ relationship and $N_I + 1$ time points so we can write $t^{(i)} \rightleftharpoons t_i$ for convenience. We now consider the mesh interval $I^{(i)}$ in the time interval $[t_{i-1}, t_i]$, then we seek a solution to the following integral equation:

$$\boldsymbol{\xi}(t_i) = \boldsymbol{\xi}(t_{i-1}) + \int_{t_{i-1}}^{t_i} \mathbf{f_d}(s, \boldsymbol{\xi}(s), \mathbf{u}(s))\, ds \tag{33}$$

Now we can rewrite Eqn. (33) to determine the defect constraints:

$$\boldsymbol{\zeta}(t_i) \rightleftharpoons \boldsymbol{\zeta}(t_{i-1}, t_i, \boldsymbol{\xi}(t_{i-1}), \boldsymbol{\xi}(t_i), \mathbf{u}(t_{i-1}), \mathbf{u}(t_i)) \tag{34a}$$

$$\boldsymbol{\zeta}(t_i) = \mathbf{0}, \qquad i = \{1, 2, \ldots, N_t\} \tag{34b}$$

$$\boldsymbol{\zeta}(t_i) = \boldsymbol{\xi}(t_i) - \boldsymbol{\xi}(t_{i-1}) - \int_{t_{i-1}}^{t_i} \mathbf{f_d}(s, \boldsymbol{\xi}(s), \mathbf{u}(s))\, ds \tag{34c}$$

We will now consider two simple Runge-Kutta schemes. For a more complete list of schemes and a more through treatment of time-marching DT methods please refer to Refs. [Bet10, Bie10, Her14a].

2.5.2 Euler Forward and Trapezoidal Defect Constraints

The Euler forward method is an explicit first-order scheme:

$$\boldsymbol{\zeta}(t_i) = \boldsymbol{\xi}(t_i) - \boldsymbol{\xi}(t_{i-1}) - h_i \mathbf{f_d}(t_{i-1}) \tag{35}$$

The trapezoidal rule is an implicit second-order scheme:

$$\boldsymbol{\zeta}(t_i) = \boldsymbol{\xi}(t_i) - \boldsymbol{\xi}(t_{i-1}) - \frac{h_i}{2}\left(\mathbf{f_d}(t_i) + \mathbf{f_d}(t_{i-1})\right) \tag{36}$$

2.5.3 Lagrange Term Quadrature

Consider the following transformation (see Ref. [Lib12, p. 87] for the assumptions required on $\mathcal{L}$):

$$\int_{t_0}^{t_f} \mathcal{L}(t, \boldsymbol{\xi}, \mathbf{u}) dt = \xi_0(t_f) \tag{37a}$$

$$\dot{\xi}_0 = \mathcal{L}(t, \boldsymbol{\xi}, \mathbf{u}), \quad \xi_0(t_0) = 0 \tag{37b}$$

This transformation adds an additional state variable ξ_0 with the dynamics naturally equivalent to $\mathcal{L}$ with an arbitrary initial value for the ordinary differential equation (ODE). The final value $\xi_0(t_f)$ can then be included in the Mayer term. For simplicity we denote the Lagrange term as:

$$\mathcal{L}(t_i) \rightleftharpoons \mathcal{L}(t_i, \boldsymbol{\xi}(t_i), \mathbf{u}(t_i)) \tag{38}$$

Now if we apply the Euler forward method to the ODE in Eqn. (37b), we arrive at the following composite quadrature method:

$$\int_{t_0}^{t_f} \mathcal{L}(t)\, dt \approx \sum_{i=0}^{N_t-1} h_i \mathcal{L}(t_i) \tag{39}$$

This is a composite quadrature method since we are using a set of points inside the time horizon to better approximate the definite integral [Hea02, p. 255]. Similarly if we apply the trapezoidal rule to the ODE in Eqn. (37b), we arrive at the following composite quadrature method:

$$\int_{t_0}^{t_f} \mathcal{L}(t)\, dt \approx \frac{1}{2} \sum_{i=0}^{N_t} h_i \left(\mathcal{L}(t_i) + \mathcal{L}(t_{i-1}) \right) \tag{40}$$

2.5.4 Remaining Problem Formulation Elements

The path and boundary constraint matrices in Eqns. (11d) and (11e) can be included in the NLP formulation in the same manner as they were for the PS method approach. As previously mentioned, there will be no continuity constraints. With the defect constraint and quadrature approaches are outlined, all the necessary numerical concepts to solve Prob. (2) with a time-marching DT method have been described.

2.5.5 Comparison

Here are a couple of comparison points between pseudospectral methods and time-marching approaches.

- At a superficial level, PS methods satisfy the dynamics by ensuring the derivatives of the approximation are sufficiently accurate while time-marching methods accomplish this task through a sufficiently accurate step size between successive state values in the ODE approximation scheme.
- The two DT approaches are called a number of terms in the literature. These various names are summarized in the following table:

similar method ↓	similar method ↓
pseudospectral	time-marching
differentiation	integration
global collocation	local collocation
	Runge-Kutta

- In general for the same level of accuracy, PS methods will involve a much smaller problem size when compared to time-marching methods.
- For some numerical comparisons between the DT methods see Ref. [Wil05].
- Another interesting comparison is with the sparsity patterns of the defect constraints for the state variables. A single-interval PS method has a fully dense state dependence blocks but only diagonal control dependence. Then a multiple-interval PS method has a block diagonal matrix structure for the states where each block corresponds to a mesh interval (see Fig. 5a in Ref. [PR14]). A time-marching method typically has an upper bidiagonal matrix structure for both the state and control components of the constraint Jacobian (see Fig. A.4 in Ref. [Her14a]). We also note that with both methods, if the derivative function is linear (i.e., $\mathbf{f_d}\left(t, \boldsymbol{\xi}(t), \mathbf{u}(t)\right) = A(t)\boldsymbol{\xi} + B(t)\mathbf{u}$), then the defect constraints are linear constraints facilitating efficient implementation.

3 Numeric Case Study

3.1 Test Problem Description

The Bryson-Denham state-constrained double integrator optimal control problem is a simple canonical test problem. A fully specified Bryson-Denham problem is formulated as:

$$\min_{u(t)} \quad \frac{1}{2}\int_0^1 [u(t)]^2\, dt \tag{41a}$$

subject to:

$$\dot{\boldsymbol{\xi}}(t) = \begin{bmatrix}\dot{x}\\ \dot{v}\end{bmatrix} = \begin{bmatrix}v(t)\\ u(t)\end{bmatrix} \tag{41b}$$

$$\boldsymbol{\xi}(0) = \begin{bmatrix}x(0)\\ v(0)\end{bmatrix} = \begin{bmatrix}0\\ 1\end{bmatrix} \tag{41c}$$

$$\boldsymbol{\xi}(1) = \begin{bmatrix}x(1)\\ v(1)\end{bmatrix} = \begin{bmatrix}0\\ -1\end{bmatrix} \tag{41d}$$

$$x(t) \leq \ell = \frac{1}{9} \tag{41e}$$

This dynamic system is analogous to moving a point mass. The closed-form solution to this problem when $0 \leq \ell \leq \frac{1}{6}$ is presented in Ref. [BH75, p. 122]:

$$u(t) = \begin{cases} -\frac{2}{3\ell}\left(1-\frac{t}{3\ell}\right) & 0 \leq t \leq 3\ell \\ 0 & 3\ell \leq t \leq 1-3\ell \\ -\frac{2}{3\ell}\left(1-\frac{1-t}{3\ell}\right) & 1-3\ell \leq t \leq 1 \end{cases} \tag{42a}$$

$$v(t) = \begin{cases} \left(1-\frac{t}{3\ell}\right)^2 & 0 \leq t \leq 3\ell \\ 0 & 3\ell \leq t \leq 1-3\ell \\ -\left(1-\frac{1-t}{3\ell}\right)^2 & 1-3\ell \leq t \leq 1 \end{cases} \tag{42b}$$

$$x(t) = \begin{cases} \ell\left(1-\left(1-\frac{t}{3\ell}\right)^3\right) & 0 \leq t \leq 3\ell \\ \ell & 3\ell \leq t \leq 1-3\ell \\ \ell\left(1-\left(1-\frac{1-t}{3\ell}\right)^3\right) & 1-3\ell \leq t \leq 1 \end{cases} \tag{42c}$$

The value of the cost functional at the optimal solution is:

$$\Psi = \frac{4}{9\ell} \tag{43}$$

This problem is well suited for studying various components of psuedospectral methods since the accuracy of the method can then be verified with the closed-form solution. The problem formulation contains many of the equations in Prob. (1): a Lagrange term, dynamic constraints, boundary constraints, and a path constraint. In addition, the closed-form solution is defined piecewise, thus is naturally suited for multiple-interval discretizations. The results can also be compared with the same problem using DT with time-marching methods in Ref. [Her14b].

3.2 Implementation and Analysis Details

A number of a case studies are performed to assess the ideal values for the user selected problem parameters, namely N_i and associated $t^{(i)}$ and $N_t^{(i)}$ elements. Both

the Legendre PS method with LGL nodes and Chebyshev PS method with CGL nodes are used.

Algorithm/Tolerances
Each problem is solved using the fmincon solver using the sqp algorithm. The default tolerances were used:

$$\text{TolCon} = 1\text{e-}6;\ \ \text{TolFun} = 1\text{e-}6;\ \ \text{TolX} = 1\text{e-}6;\ \ \text{TolConSQP} = 1\text{e-}6;$$

Performance Metrics
Absolute error e will be calculated with respect to the closed-form solution in Eqn. (42) and closed-form objective function value in Eqn. (43):

$$\text{absolute error} = |\text{actual} - \text{approx}| \tag{44}$$

The accuracy of a trajectory will be assessed with the maximum absolute error on the interval. A final metric will be the total CPU time[1] to assess the expected trade-offs between accuracy and computation time.

Initial Guess
Initial guesses are important when using direct methods of optimal control [Wil05]. Here a linear initial trajectory between the state boundary conditions was used and an all zero initial control guess.

3.3 Summary of Case Study Results

The results for 15 numerical studies are shown in Table 1 with LGL nodes and Table 2 with CGL nodes. Select solutions and error plots from the tables are shown in Figs. 2–3.

- Tests 7–9 use 3 mesh intervals defined exactly at the piecewise solution points in Eqn. (42). We are unlikely to know these points a priori so these solutions do not reflect the general performance of PS methods. Test 7 uses polynomial orders equivalent to the highest-order polynomial solution in Eqn. (42). Therefore, the approximations in Sec. 2.1 are expected to be exact. This is shown with the max errors of both the state and control near machine precision and the objective function exactly valued at 4 (see Fig.3a also well). Increase N_t on the intervals introduces additional numerical errors even though we expect the polynomial approximations to be exact, i.e., we are using more collocation points than are needed to approximate the true solution.

- The intent of tests 1–3 was to show the effect of increasing N_t with a single mesh interval. We see slow convergence to the closed-form solution. Fig. 2 shows test 1 and test 3 solutions.

- The whole collection of tests (excluding 7–9) demonstrate that a *large number of mesh intervals with medium-order polynomials* ($3 \leq N_t \leq 6$) are best of determining the solution of this problem. Test 11 seems to have a reasonable mesh specification with small errors and a reasonable computation time.

- There seems to be little difference between the results using either LGL or CGL nodes.

[1]Tests were performed with MATLAB r2014a on an i5-2500K 4.2 GHz CPU, 16GB RAM, Win8.1 computer.

Table 1 Results for different meshes using LGL nodes.

Test	$\mathbf{t}$	$N_t^{(i)}$	e_Ψ	$\max e_x$	$\max e_v$	$\max e_u$	t_{CPU} (s)
1	$[0,1]$	4	2×10^{-2}	1×10^{-3}	2×10^{-2}	5×10^{-1}	0.05
2	$[0,1]$	9	4×10^{-2}	2×10^{-3}	2×10^{-2}	5×10^{-1}	0.17
3	$[0,1]$	19	4×10^{-3}	3×10^{-4}	5×10^{-3}	2×10^{-1}	2.99
4	$[0,\frac{1}{2},1]$	**3**	8×10^{-2}	8×10^{-3}	8×10^{-2}	2×10^{0}	0.11
5	$[0,\frac{1}{2},1]$	**5**	5×10^{-3}	7×10^{-4}	8×10^{-3}	3×10^{-1}	0.53
6	$[0,\frac{1}{2},1]$	**9**	6×10^{-3}	5×10^{-4}	8×10^{-3}	3×10^{-1}	2.82
7	$[0,3\ell,1-3\ell,1]$	$[3,1,3]$	1×10^{-15}	1×10^{-16}	4×10^{-16}	4×10^{-15}	0.16
8	$[0,3\ell,1-3\ell,1]$	$[5,1,5]$	5×10^{-10}	2×10^{-9}	2×10^{-8}	1×10^{-6}	0.37
9	$[0,3\ell,1-3\ell,1]$	**5**	4×10^{-12}	3×10^{-8}	3×10^{-7}	1×10^{-5}	2.40
10	$\mathsf{linspace}(0,1,6)$	**3**	2×10^{-2}	3×10^{-3}	3×10^{-2}	6×10^{-1}	0.87
11	$\mathsf{linspace}(0,1,6)$	**5**	4×10^{-4}	1×10^{-4}	2×10^{-3}	2×10^{-1}	7.35
12	$\mathsf{linspace}(0,1,6)$	**9**	4×10^{-4}	1×10^{-4}	1×10^{-3}	1×10^{-1}	40.35
13	$\mathsf{linspace}(0,1,11)$	**3**	2×10^{-3}	2×10^{-4}	6×10^{-3}	3×10^{-1}	8.81
14	$\mathsf{linspace}(0,1,11)$	**5**	3×10^{-5}	2×10^{-4}	2×10^{-3}	8×10^{-2}	51.60
15	$\mathsf{linspace}(0,1,11)$	**9**	4×10^{-5}	3×10^{-5}	3×10^{-4}	6×10^{-2}	269.39

Table 2 Results for different meshes using CGL nodes.

Test	$\mathbf{t}$	$N_t^{(i)}$	e_Ψ	$\max e_x$	$\max e_v$	$\max e_u$	t_{CPU} (s)
1	$[0,1]$	4	2×10^{-2}	1×10^{-3}	2×10^{-2}	5×10^{-1}	0.05
2	$[0,1]$	9	5×10^{-2}	2×10^{-3}	3×10^{-2}	5×10^{-1}	0.17
3	$[0,1]$	19	5×10^{-3}	4×10^{-4}	6×10^{-3}	3×10^{-1}	3.14
4	$[0,\frac{1}{2},1]$	**3**	5×10^{-2}	7×10^{-3}	8×10^{-2}	2×10^{0}	0.11
5	$[0,\frac{1}{2},1]$	**5**	5×10^{-3}	5×10^{-4}	6×10^{-3}	4×10^{-1}	0.68
6	$[0,\frac{1}{2},1]$	**9**	6×10^{-3}	5×10^{-4}	7×10^{-3}	3×10^{-1}	2.87
7	$[0,3\ell,1-3\ell,1]$	$[3,1,3]$	4×10^{-15}	1×10^{-16}	8×10^{-16}	7×10^{-15}	0.17
8	$[0,3\ell,1-3\ell,1]$	$[5,1,5]$	7×10^{-15}	2×10^{-9}	1×10^{-8}	5×10^{-7}	0.46
9	$[0,3\ell,1-3\ell,1]$	**5**	2×10^{-12}	2×10^{-8}	2×10^{-7}	6×10^{-6}	2.41
10	$\mathsf{linspace}(0,1,6)$	**3**	1×10^{-2}	3×10^{-3}	3×10^{-2}	6×10^{-1}	0.88
11	$\mathsf{linspace}(0,1,6)$	**5**	4×10^{-4}	7×10^{-5}	2×10^{-3}	1×10^{-1}	7.76
12	$\mathsf{linspace}(0,1,6)$	**9**	4×10^{-4}	1×10^{-4}	1×10^{-3}	1×10^{-1}	40.59
13	$\mathsf{linspace}(0,1,11)$	**3**	1×10^{-3}	2×10^{-4}	6×10^{-3}	3×10^{-1}	8.88
14	$\mathsf{linspace}(0,1,11)$	**5**	3×10^{-5}	1×10^{-4}	1×10^{-3}	7×10^{-2}	56.79
15	$\mathsf{linspace}(0,1,11)$	**9**	6×10^{-5}	3×10^{-5}	4×10^{-4}	6×10^{-2}	258.26

Acknowledgments

I would like to thank Assistant Professor James Allison[2] at the University of Illinois at Urbana-Champaign who provided much assistance when preparing this report.

References

[Bec11] V M Becerra. *PSOPT Optimal Control Solver User Manual*, release 3 build 2011-07-28 edition, July 2011. url: https://psopt.googlecode.com/files/PSOPT_Manual_R3.pdf

[Bet10] J T Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, 2010. doi: 10.1137/1.9780898718577

[BGa10] V M Becerra and R K H Galvão. Um Tutorial Sobre Métodos Pseudo-Espectrais Para Controle Ótimo Computacional. *Sba Controle & Automação*, 21(3):224–244, May-June 2010. doi: 10.1590/S0103-17592010000300002

[BH75] A E Bryson and Y C Ho. *Applied Optimal Control: Optimization, Estimation and Control*. Taylor & Francis, 1975.

[Bie07] L T Biegler. An Overview of Simultaneous Strategies for Dynamic Optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, November 2007. doi: 10.1016/j.cep.2006.06.021

[Bie10] L T Biegler. *Nonlinear Programming: Concepts, Algorithms and Applications to Chemical Engineering*. MOS-SIAM Series on Optimization, 2010. doi: 10.1137/1.9780898719383

[FR02] F Fahroo and I M Ross. Direct Trajectory Optimization by a Chebyshev Pseudospectral Method. *AIAA Journal Of Guidance, Control, and Dynamics*, 25(1):160–166, January-February 2002. doi: 10.2514/2.4862

[FR08] F Fahroo and I M Ross. Advances in Pseudospectral Methods for Optimal Control. In *AIAA 2008 Guidance, Navigation and Control Conference and Exhibit*, number AIAA 2008-7309, Honolulu, HI, USA, August 2008. doi: 10.2514/6.2008-7309

[Hea02] M T Heath. *Scientific Computing: An Introductory Survey*. McGraw Hill, 2nd edition, 2002.

[Her14a] D R Herber. Dynamic System Design Optimization of Wave Energy Converters Utilizing Direct Transcription. M.S. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, May 2014. url: http://hdl.handle.net/2142/49463

[Her14b] D R Herber. Solving Optimal Control Problems using Simscape Models for State Derivatives. Technical Report UIUC-ESDL-2014-01, Engineering System Design Lab, July 2014. url: http://hdl.handle.net/2142/50015

[2] http://www.mathworks.com/matlabcentral/fileexchange/authors/157997

[Lib12] D Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012.

[PR14] M A Patterson and A V Rao. $\mathbb{GPOPS}-\mathbb{II}$: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using *hp*-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Transactions on Mathematical Software*, 41(1), October 2014. doi: 10.1145/2558904

[Rao12] A V Rao. Pseudospectral Methods for Optimal Control: Theory and Practice. Online, November 2012.

[RE10] P E Rutquist and M M Edvall. *PROPT - Matlab Optimal Control Software*. TOMLAB Optimization, Pullman, WA, USA, April 2010.

[STW11] J Shen, T Tang, and L-L Wang. *Spectral Methods: Algorithms, Analysis and Applications*. Springer, 2011. doi: 10.1007/978-3-540-71041-7

[Tre00] L N Trefethen. *Spectral Methods in MATLAB*. SIAM, 2000. doi: 10.1137/1.9780898719598

[Wal06] J Waldvogel. Fast Construction Of The Fejér And Clenshaw-Curtis Quadrature Rules. *BIT Numerical Mathematics*, 46(1):195–202, March 2006. doi: 10.1007/s10543-006-0045-4

[Wil05] P Williams. A Comparison of Differentiation and Integration Based Direct Transcription Methods. In *AAS/AIAA 2005 Space Flight Mechanics Meetings*, volume 120, pages 389–408, Copper Mountain, CO, USA, January 2005. Paper AAS 05-128.

A Case Studies

A.1 Select Command Window Outputs

Test 3 with LGL nodes and scaling

Iter	F-count	f(x)	Feasibility	Steplength	Norm of step	First-order optimality
0	61	0.000000e+00	1.000e+00			9.581e-07
1	122	4.182098e+00	7.637e-08	1.000e+00	7.490e-01	8.742e+00
2	190	4.141263e+00	7.683e-08	8.235e-02	1.123e-01	8.878e+00
3	258	4.110792e+00	8.208e-08	8.235e-02	9.393e-02	6.532e+00
4	325	4.089447e+00	7.272e-08	1.176e-01	9.340e-02	6.514e+00
5	391	4.069116e+00	5.627e-08	1.681e-01	1.021e-01	5.566e+00
6	458	4.042053e+00	4.678e-08	1.176e-01	7.704e-02	4.720e+00
7	520	4.028904e+00	5.750e-09	7.000e-01	2.059e-01	1.718e+00
8	581	3.997462e+00	9.402e-09	1.000e+00	9.699e-02	4.633e-01
9	652	3.997136e+00	6.779e-09	2.825e-02	7.586e-03	1.344e-01
10	713	3.996297e+00	1.466e-09	1.000e+00	8.680e-03	1.170e-02
11	783	3.996297e+00	1.381e-09	4.035e-02	1.920e-04	1.162e-02
12	847	3.996296e+00	8.598e-10	3.430e-01	5.120e-04	9.202e-03
13	916	3.996295e+00	8.273e-10	5.765e-02	7.767e-04	1.045e-02
14	986	3.996294e+00	7.561e-10	4.035e-02	2.703e-04	1.167e-02
15	1053	3.996294e+00	6.370e-10	1.176e-01	1.713e-04	1.164e-02
16	1117	3.996293e+00	3.261e-10	3.430e-01	6.624e-04	5.821e-03
17	1185	3.996293e+00	2.115e-10	8.235e-02	3.653e-04	5.124e-03
18	1246	3.996292e+00	1.128e-11	1.000e+00	3.737e-04	3.097e-04
19	1307	3.996292e+00	1.952e-12	1.000e+00	2.152e-05	1.388e-05
20	1371	3.996292e+00	1.120e-12	3.430e-01	6.864e-06	9.026e-06
21	1432	3.996292e+00	1.599e-14	1.000e+00	2.539e-06	1.071e-07

```
Local minimum found that satisfies the constraints.

Elapsed time is 3.114216 seconds.
```

Test 3 with LGL nodes and scaling

Test 3 with LGL nodes and no scaling

Iter	F-count	f(x)	Feasibility	Steplength	Norm of step	First-order optimality
0	61	0.000000e+00	1.000e+00			5.988e-10
1	122	4.440695e+00	2.237e-07	1.000e+00	1.453e+01	5.188e+00
2	183	4.419303e+00	3.818e-09	1.000e+00	1.476e-01	6.131e-02
3	244	4.325927e+00	1.646e-08	1.000e+00	7.143e-01	5.436e-02
4	305	4.144487e+00	1.564e-08	1.000e+00	2.479e+00	3.674e-02
5	366	4.121958e+00	7.728e-09	1.000e+00	8.743e-01	3.042e-02
6	427	4.109372e+00	1.337e-08	1.000e+00	3.636e-01	2.649e-02
7	488	4.105732e+00	2.811e-10	1.000e+00	9.628e-02	1.722e-02
8	549	4.089985e+00	2.118e-09	1.000e+00	4.036e-01	1.512e-02
9	610	4.060340e+00	1.048e-08	1.000e+00	1.552e+00	1.347e-02
10	671	4.059341e+00	2.062e-10	1.000e+00	8.235e-02	1.341e-02
11	732	4.055693e+00	4.896e-09	1.000e+00	3.272e-01	1.312e-02
12	793	4.052156e+00	1.153e-09	1.000e+00	2.833e-01	1.274e-02
13	854	4.043426e+00	2.217e-09	1.000e+00	7.581e-01	1.155e-02
14	915	4.035815e+00	5.623e-09	1.000e+00	7.955e-01	1.387e-02
15	976	4.031338e+00	1.001e-08	1.000e+00	5.709e-01	1.152e-02
16	1037	4.029493e+00	2.673e-09	1.000e+00	2.265e-01	1.042e-02
17	1098	4.027423e+00	3.812e-09	1.000e+00	1.972e-01	1.226e-02
18	1159	4.023017e+00	4.228e-09	1.000e+00	3.884e-01	1.322e-02
19	1220	4.014998e+00	3.688e-09	1.000e+00	8.026e-01	1.137e-02
20	1281	4.005079e+00	4.906e-09	1.000e+00	1.243e+00	1.271e-02
21	1342	3.999058e+00	5.680e-09	1.000e+00	9.945e-01	1.040e-02
22	1403	3.997552e+00	3.279e-09	1.000e+00	2.982e-01	5.571e-03
23	1464	3.997276e+00	1.474e-09	1.000e+00	8.647e-02	4.589e-03
24	1525	3.997041e+00	7.742e-10	1.000e+00	6.581e-02	3.446e-03
25	1586	3.996665e+00	2.516e-10	1.000e+00	7.044e-02	2.412e-03
26	1647	3.996390e+00	5.434e-10	1.000e+00	8.808e-02	1.134e-03

```
   27    1708    3.996302e+00   3.210e-10   1.000e+00   8.893e-02   3.904e-04
   28    1769    3.996293e+00   5.595e-11   1.000e+00   3.081e-02   1.640e-04
   29    1830    3.996293e+00   2.836e-11   1.000e+00   3.790e-03   1.501e-04
   30    1891    3.996293e+00   1.701e-11   1.000e+00   9.169e-04   1.450e-04
   31    1952    3.996292e+00   2.365e-11   1.000e+00   1.923e-03   7.148e-05
   32    2013    3.996292e+00   1.610e-11   1.000e+00   1.489e-03   2.976e-05
   33    2074    3.996292e+00   2.446e-11   1.000e+00   1.060e-03   8.578e-06
   34    2135    3.996292e+00   1.545e-12   1.000e+00   2.881e-04   1.519e-06
   35    2196    3.996292e+00   1.644e-12   1.000e+00   4.163e-05   9.324e-08

Local minimum found that satisfies the constraints.

Elapsed time is 4.959984 seconds.
```

Test 3 with LGL nodes and no scaling

Test 7 with LGL nodes and scaling

```
                                                        Norm of First-order
 Iter F-count            f(x) Feasibility  Steplength        step  optimality
    0       31    0.000000e+00   3.333e-01                          1.987e-06
    1       62    4.000000e+00   2.723e-09   1.000e+00   5.863e-01   2.401e+01
    2       93    4.000000e+00   3.553e-15   1.000e+00   1.147e-09   1.242e-07

Local minimum found that satisfies the constraints.

Elapsed time is 0.163738 seconds.
```

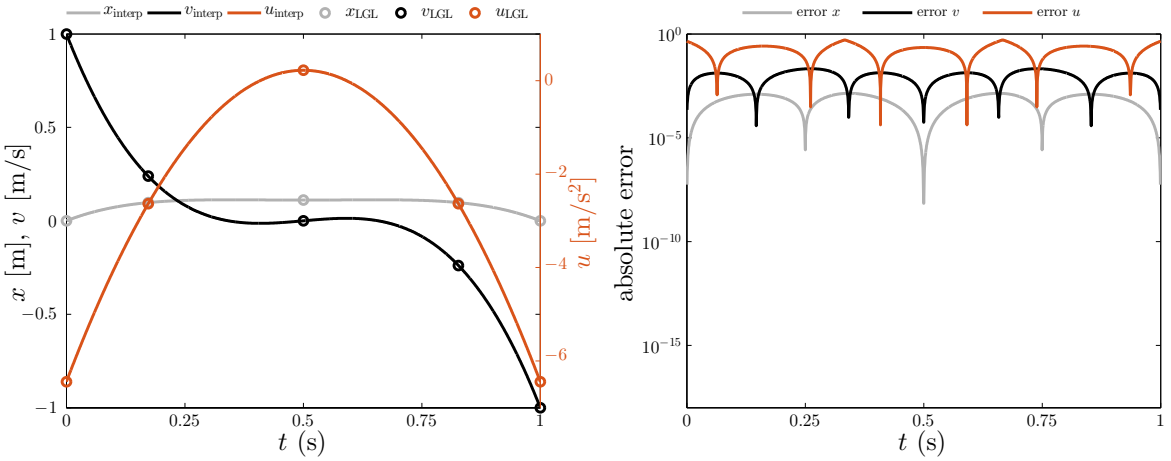
Test 7 with LGL nodes and scaling

Test 11 with LGL nodes and scaling

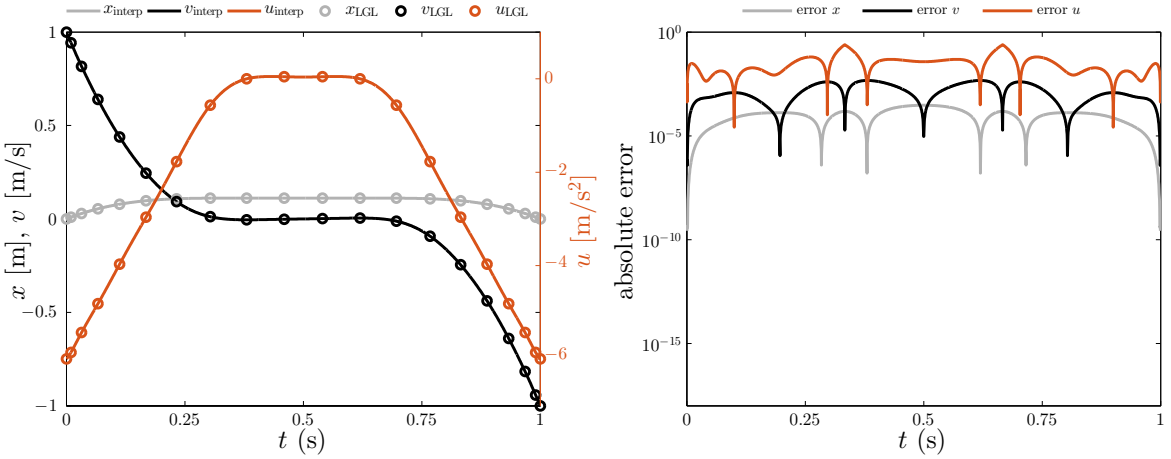
```
                                                        Norm of First-order
 Iter F-count            f(x) Feasibility  Steplength        step  optimality
    0       91    0.000000e+00   2.000e-01                          6.614e-07
    1      182    6.004637e+00   7.290e-09   1.000e+00   9.052e-01   1.538e+01
    2      274    5.946535e+00   5.962e-09   7.000e-01   6.715e-01   1.441e+01
    3      367    5.463725e+00   2.080e-09   4.900e-01   3.717e-01   6.665e+00
    4      461    5.448271e+00   3.466e-09   3.430e-01   6.082e-01   7.539e+00
    5      553    5.238700e+00   2.108e-09   7.000e-01   4.548e-01   7.038e+00
    6      647    4.937073e+00   5.122e-09   3.430e-01   4.584e-01   4.070e+00
    7      742    4.809083e+00   3.166e-09   2.401e-01   2.899e-01   4.585e+00
    8      836    4.657508e+00   4.109e-09   3.430e-01   2.472e-01   3.820e+00
    9      931    4.513987e+00   3.917e-09   2.401e-01   2.648e-01   2.514e+00
   10     1022    4.001174e+00   4.705e-10   1.000e+00   1.422e-01   5.090e-01
   11     1120    4.001110e+00   4.157e-10   8.235e-02   4.794e-03   4.645e-01
   12     1214    4.001038e+00   2.192e-10   3.430e-01   7.169e-03   3.226e-01
   13     1309    4.000880e+00   1.275e-10   2.401e-01   1.140e-02   1.838e-01
   14     1403    4.000818e+00   1.199e-10   3.430e-01   6.289e-03   1.565e-01
   15     1496    4.000669e+00   1.063e-10   4.900e-01   9.843e-03   1.209e-01
   16     1592    4.000660e+00   9.271e-11   1.681e-01   5.269e-03   9.754e-02
   17     1683    4.000406e+00   2.435e-11   1.000e+00   6.498e-03   3.082e-02
   18     1774    4.000370e+00   1.091e-11   1.000e+00   1.816e-03   4.188e-03
   19     1865    4.000370e+00   1.214e-12   1.000e+00   1.199e-04   8.229e-04
   20     1956    4.000370e+00   3.950e-13   1.000e+00   2.361e-05   4.037e-05
   21     2047    4.000370e+00   1.693e-15   1.000e+00   8.035e-14   5.761e-06

Local minimum found that satisfies the constraints.

Elapsed time is 7.681312 seconds.
```

Test 11 with LGL nodes and scaling

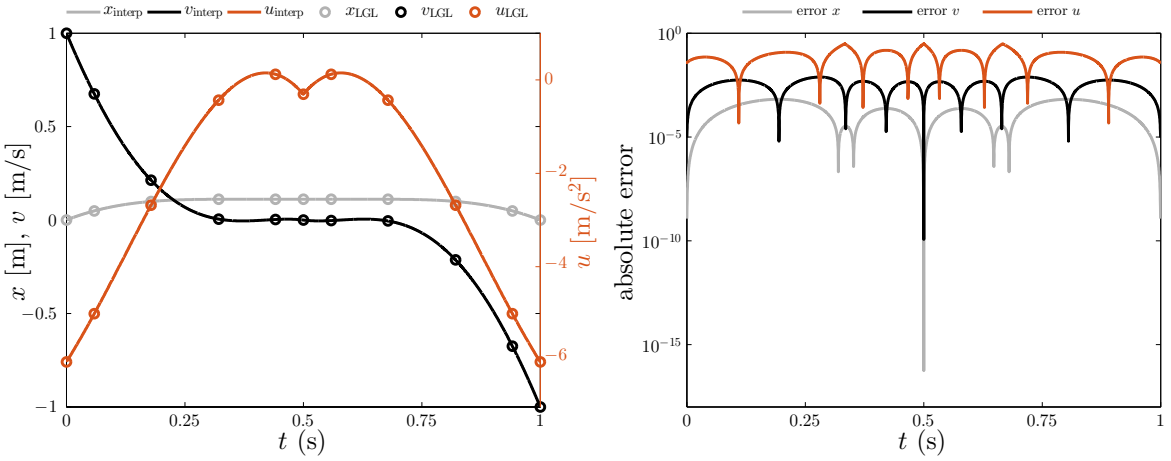
A.2 Select Solution and Error Plots



(a) Test 1, LGL nodes.

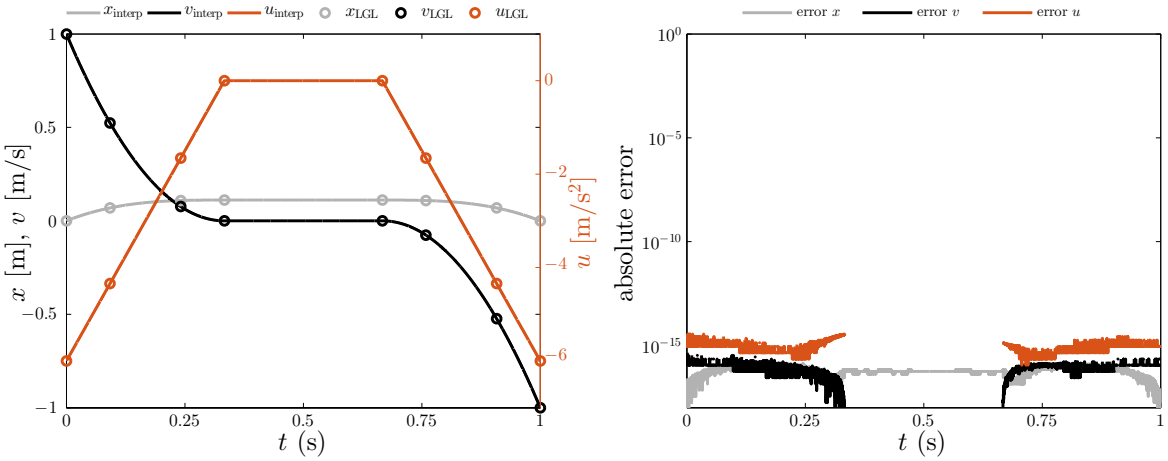


(b) Test 3, LGL nodes.

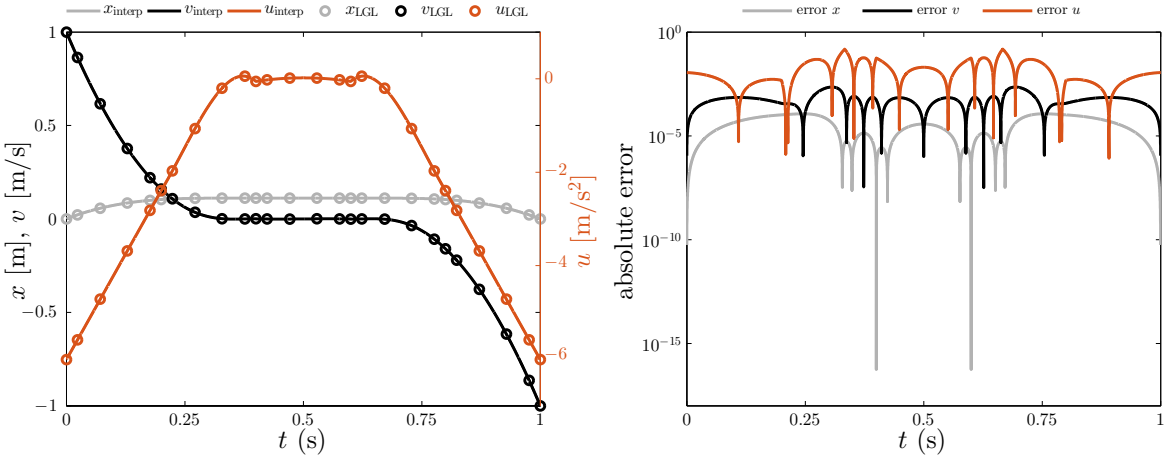


(c) Test 5, LGL nodes.

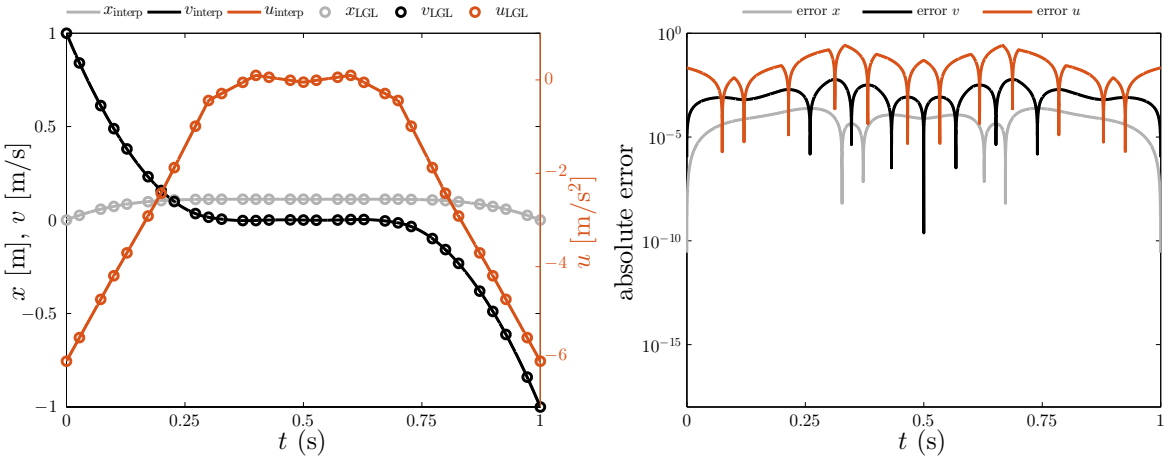
Figure 2 Solution and error plots for tests $\{1, 3, 5\}$ with LGL nodes.



(a) Test 7, LGL nodes.

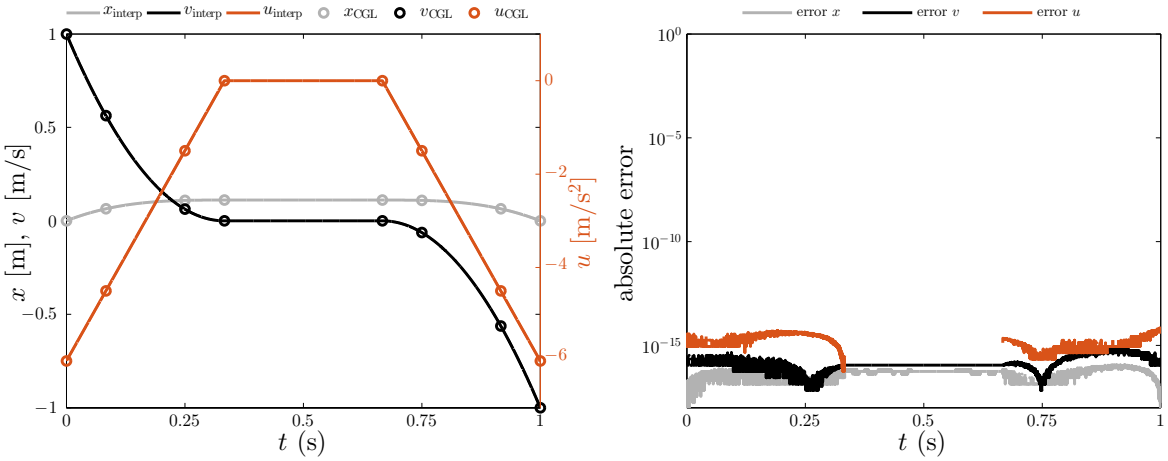


(b) Test 11, LGL nodes.

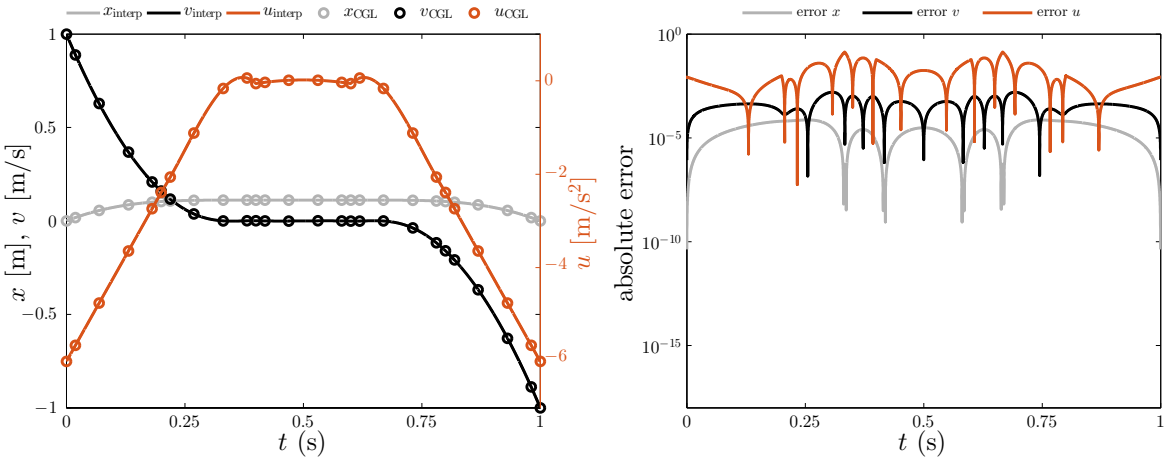


(c) Test 13, LGL nodes.

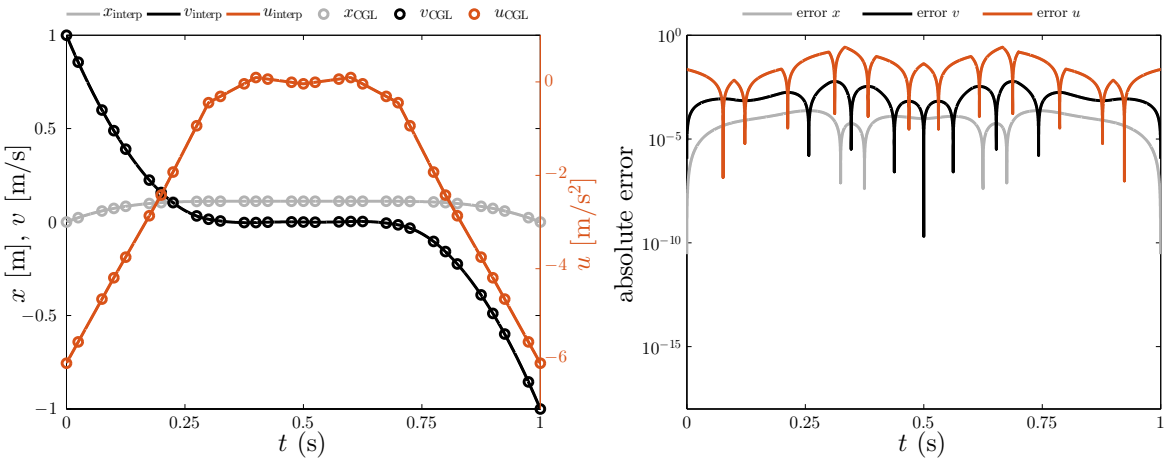
Figure 3 Solution and error plots for tests $\{7, 11, 13\}$ with LGL nodes.



(a) Test 7, CGL nodes.



(b) Test 11, CGL nodes.

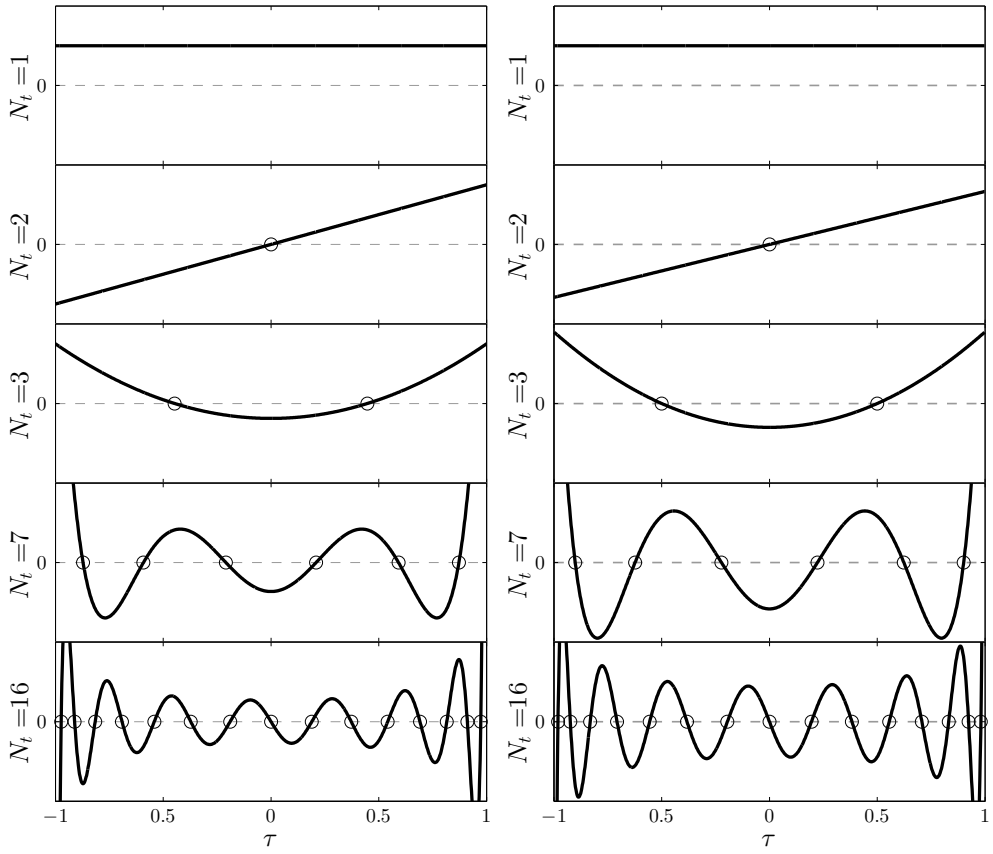


(c) Test 13, CGL nodes.

Figure 4 Solution and error plots for tests $\{7, 11, 13\}$ with CGL nodes.

B Visualizations for Specific Pseudospectral Implementations

This appendix contains a number of figures devoted to explaining the various aspects of the specific pseudospectral methods outlined in Secs. 2.3 and 2.4. Many of the figures are similar to the ones found in Refs. [BGa10, Bec11, Rao12] so please refer to them for further analysis.



(a) Inner LGL nodes based on $\dot{L}_{N_t}(\tau)$. **(b)** Inner CGL nodes based on $\dot{T}_{N_t}(\tau)$.

Figure 5 LGL and CGL inner node locations from the roots of polynomials.

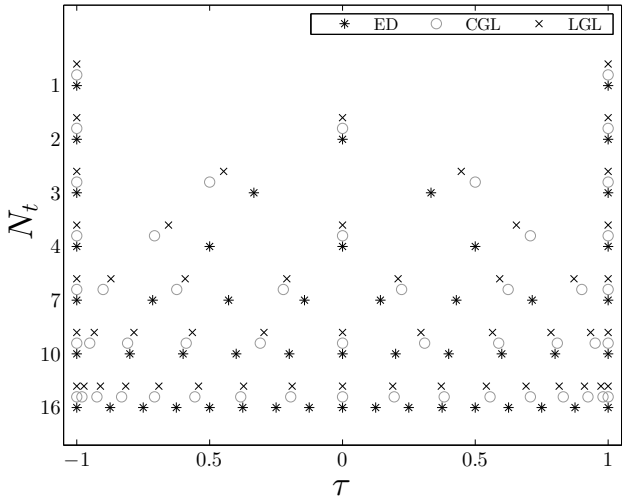


Figure 6 ED, CGL, and LGL nodes for various values of N_t.

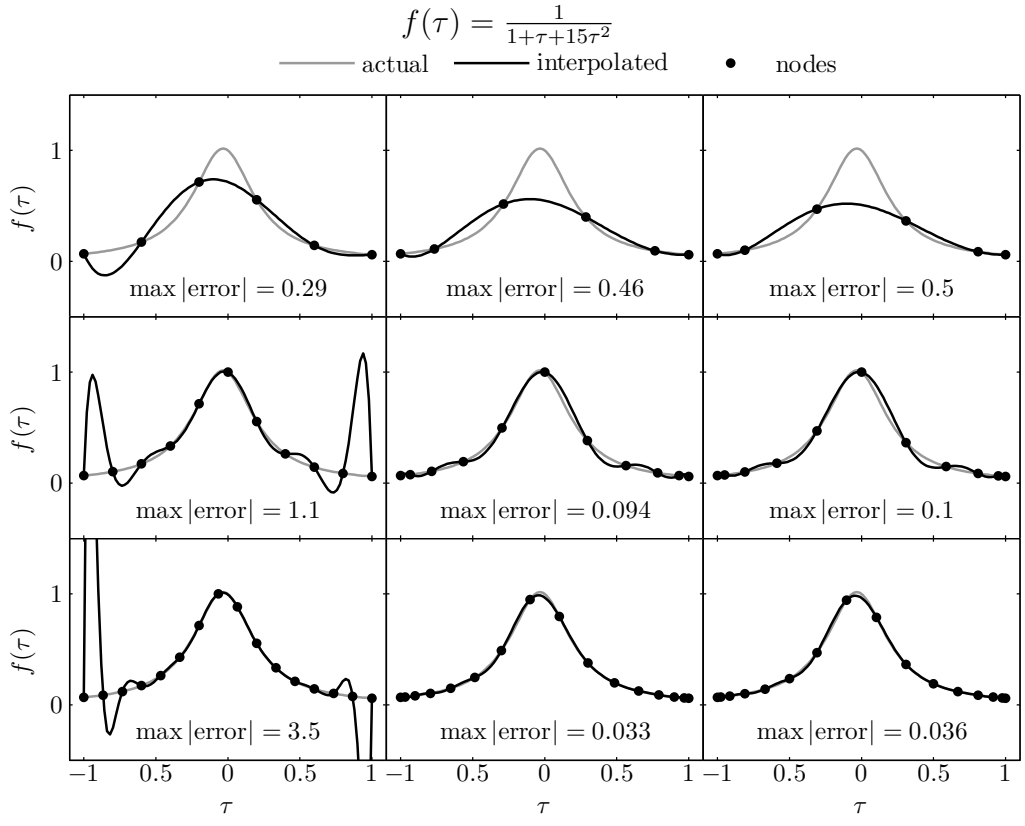


Figure 7 Lagrange polynomial interpolation with ED, LGL, and CGL nodes for various values of N_t.

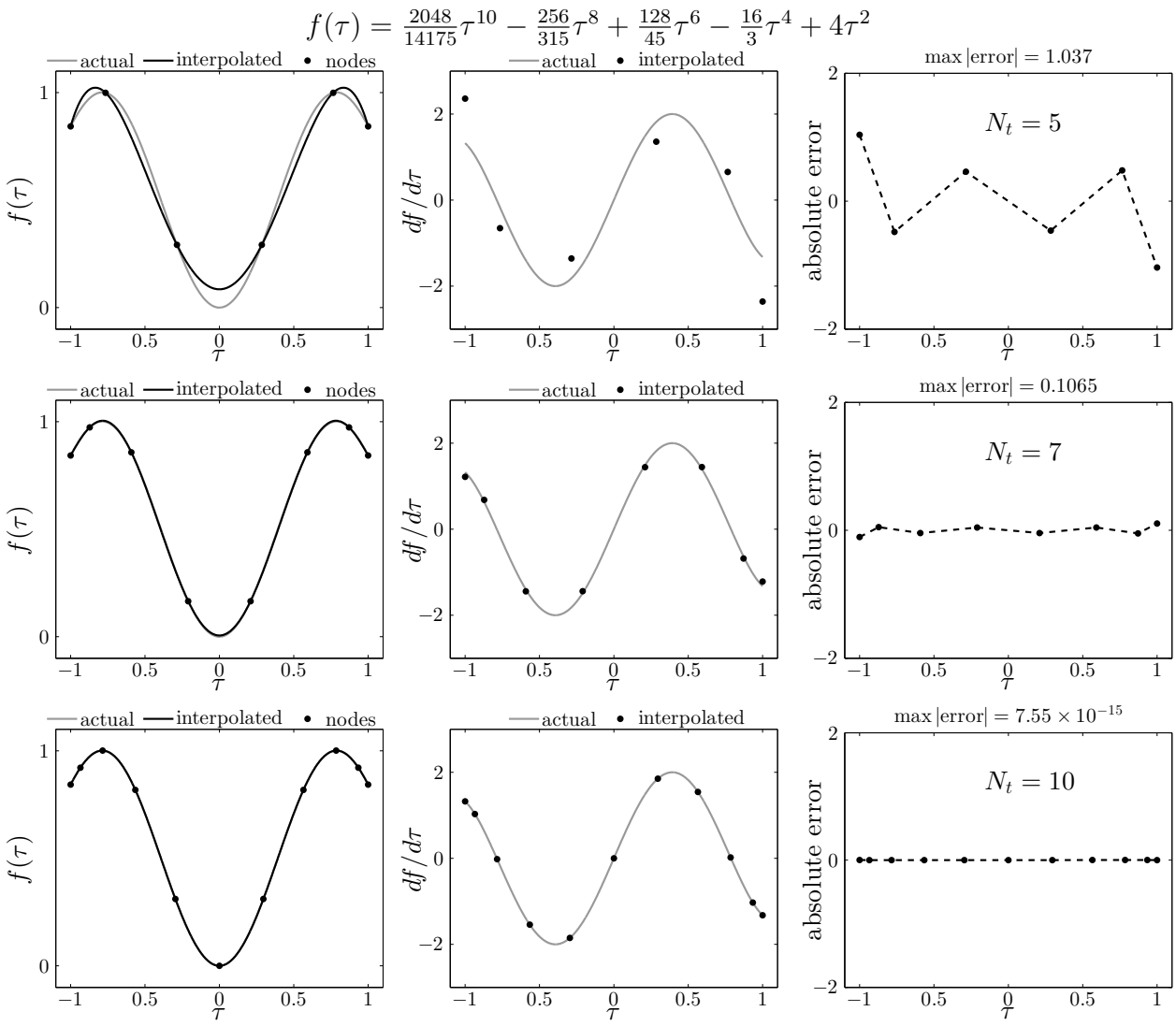
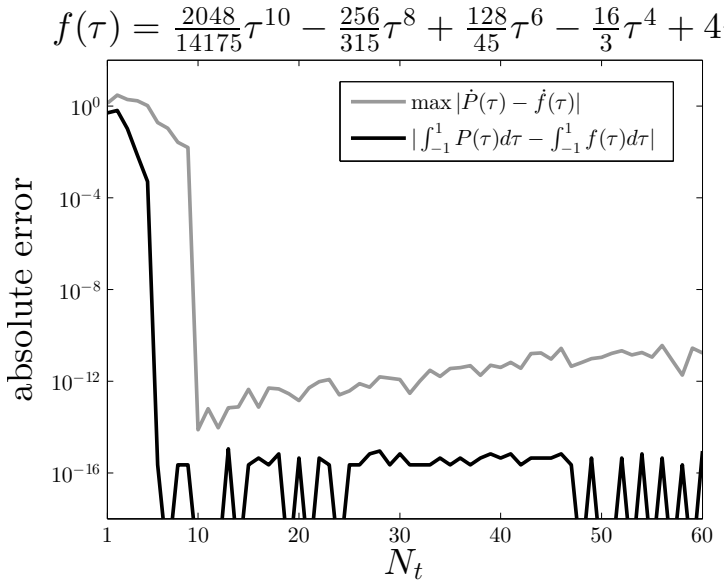
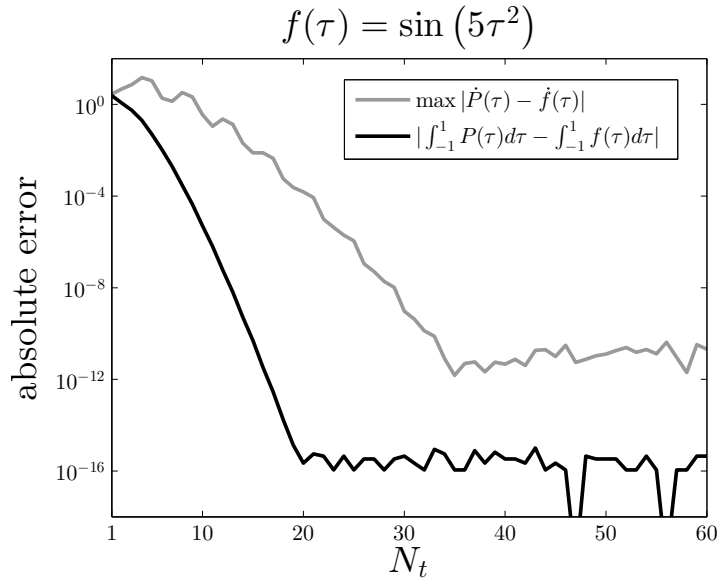


Figure 8 Differentiation error using Lagrange interpolation with LGL nodes for various values of N_t.



(a) Polynomial example.



(b) Nonpolynomial example.

Figure 9 Convergence behavior for definite integral and derivative approximations using Lagrange interpolation with LGL nodes.

C Supplementary MATLAB® Code

The code used in the numeric case studies is available at:
`http://www.mathworks.com/matlabcentral/fileexchange/51104`

The script PS_main.m can solve the test problem for each one of the meshes defined in Tables 1 and 2 with the study variable. If study = 0 then a user defined mesh can be provided in the switch statement. To change the pseudospectral method to be either based on LGL or CGL nodes, modify either p.method = 'LGL' or p.method = 'CGL'. Scaling of the variables and modifying ℓ in Prob. (41) can also be changed in the script (but recall that $0 \leq \ell \leq \frac{1}{6}$ for the solution in Eqn. (42) to be valid).

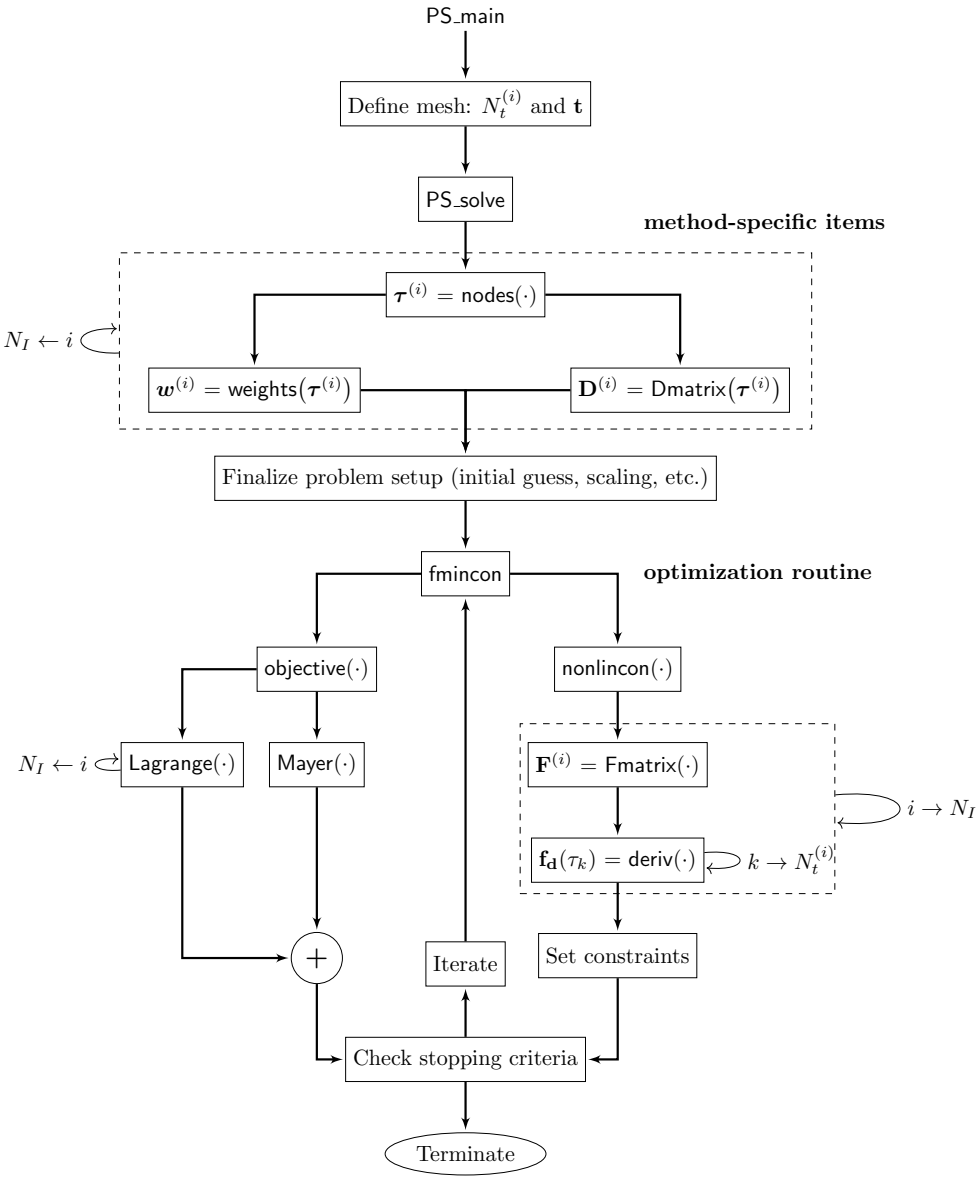


Figure 10 Algorithm flowchart of supplementary MATLAB® code.